Pair cascades in active galactic nuclei

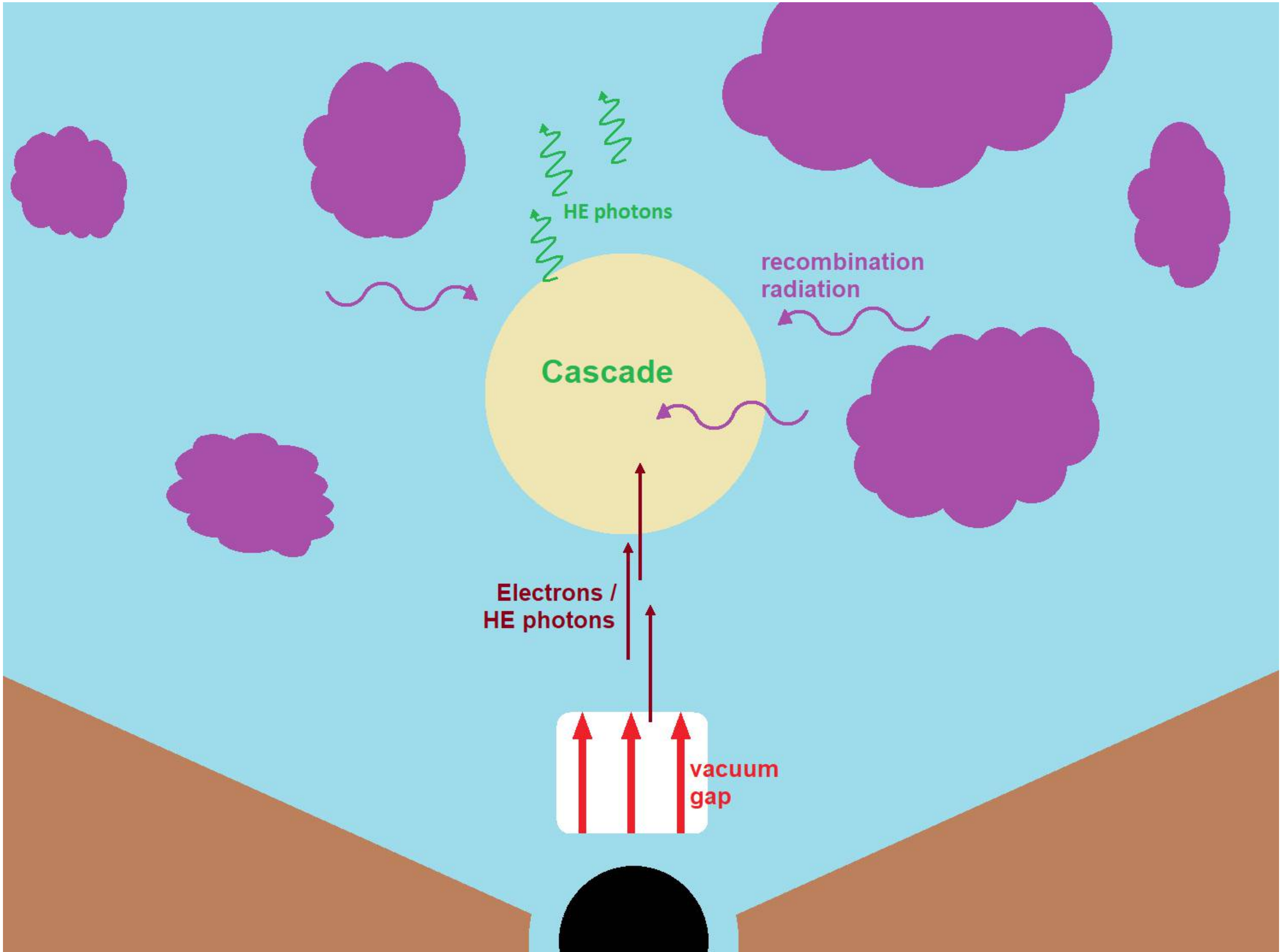
Talk by Christoph Wendel

In collaboration with Amit Shukla and Karl Mannheim

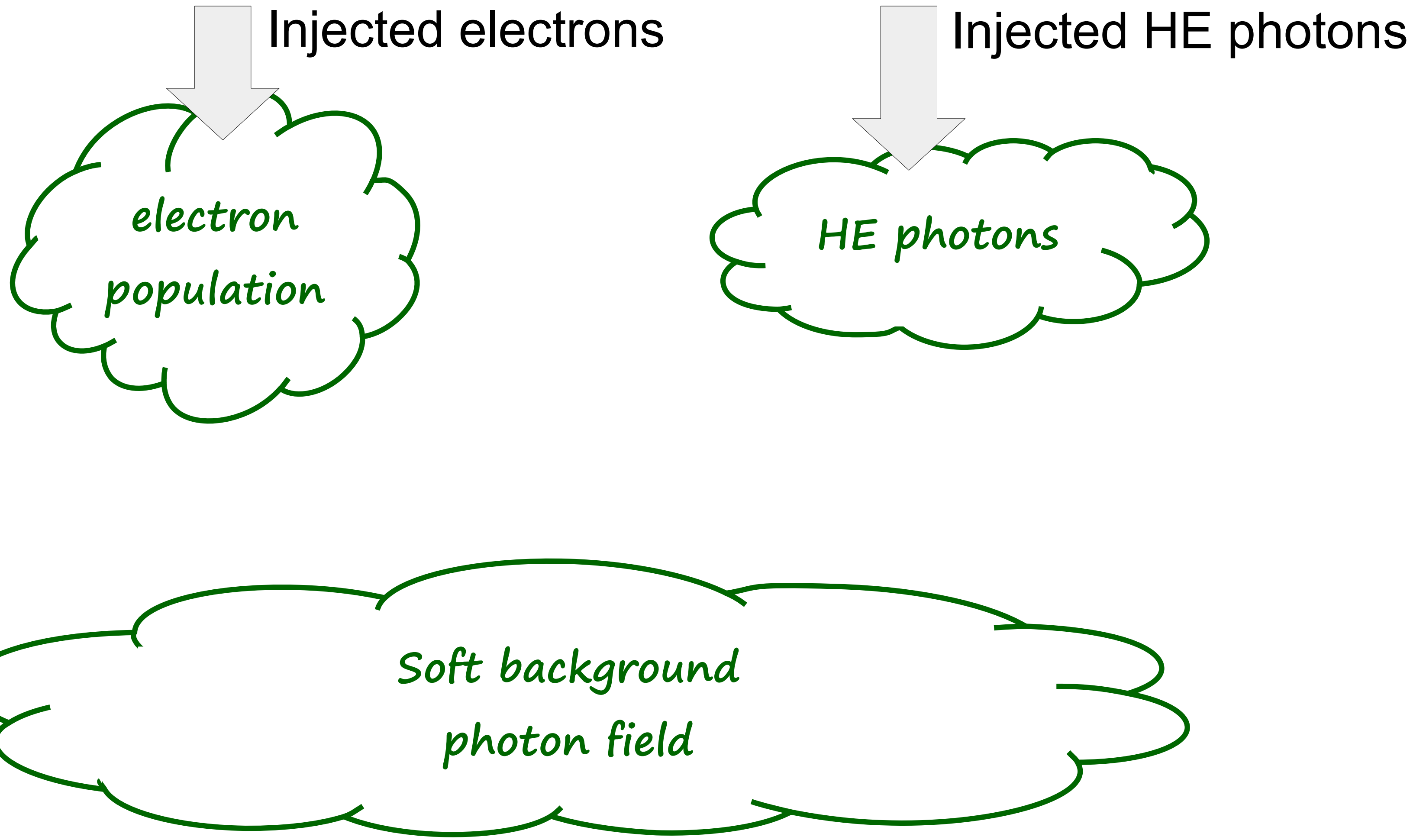
DPG spring meeting, Würzburg, 22.03.2018

```
## Considering various inverse-Compton-scattering and pair-production formulae

import numpy as np
from scipy import integrate
from GeneralDefinitions import *

c = 290000000.0      # The velocity of light in units m/s
h = 6.6*10**(-34)    # The Planck's Constant in units J*s
me = 9.1*10**(-31)   # The electron's mass in units kg
kB = 1.4*10**(-23)   # The Boltzmann's Constant in units J/K

def n0Planck(x,PlanckTemperature):# This gives the spectral number-density of the soft photons in units 1/m^3.
    return (8*np.pi*me**3*c**3*x**2) / ( h**3 * (np.exp(x*me*c**2/(kB*PlanckTemperature)) - 1) )

x0Delta = 30*Theta                # The situation of the Delta-peak. INPUT VALUE!
def n0Delta(x):                   # This gives the number-density of the soft photons in units 1/m^3.
    A = 10**(27)                  # The coefficient for n0. INPUT VALUE!
    return A

if Usedn0 == n0Delta:
    x0 = x0Delta                  # This makes the following code easier.
    def COfA1(ga,gaP,n0):         # Defining this COfA1 separately is a trick to bypass the integration.
        return IntegrandOfA1(x0,ga,gaP,n0)
else:
    x0 = x0NonDelta               # This makes the following code easier.
    def COfA1(ga,gaP,n0):         # Equation A1.
        LowerIntBorder = max(x1,EAstOfA3(ga,gaP)/ga)    # Definition of lower integration border.
        if LowerIntBorder >= x0:
            LowerIntBorder=x0     # Prevent the lower integration border from exceeding the upper border.
        COfA1Result, COfA1Error = integrate.quad(IntegrandOfA1, LowerIntBorder, x0, args=(ga,gaP,n0))
        return COfA1Result

ValuesForgammaP = np.logspace(np.log10(gaPminOf6(x0,gamma)),NumOfDecadesOfgamma,NumOfDecadesOfgamma*10+1)
ValuesForCOfA1 = [COfA1(gamma,i,Usedn0) for i in ValuesForgammaP.Lin]   # Evaluate the integral.
pl.figure(figsize=(12, 9), num="Inverse-Compton energy spectral density versus final electron energy")
pl.plot(ValuesForgammaP,ValuesForCOfA1, label=LabelForIC)               # Plot the function.
ax = pl.gca()                                                           # Get current axes.
ax.xaxis.set_tick_params(labelsize=16)                                  # Set ticks
```

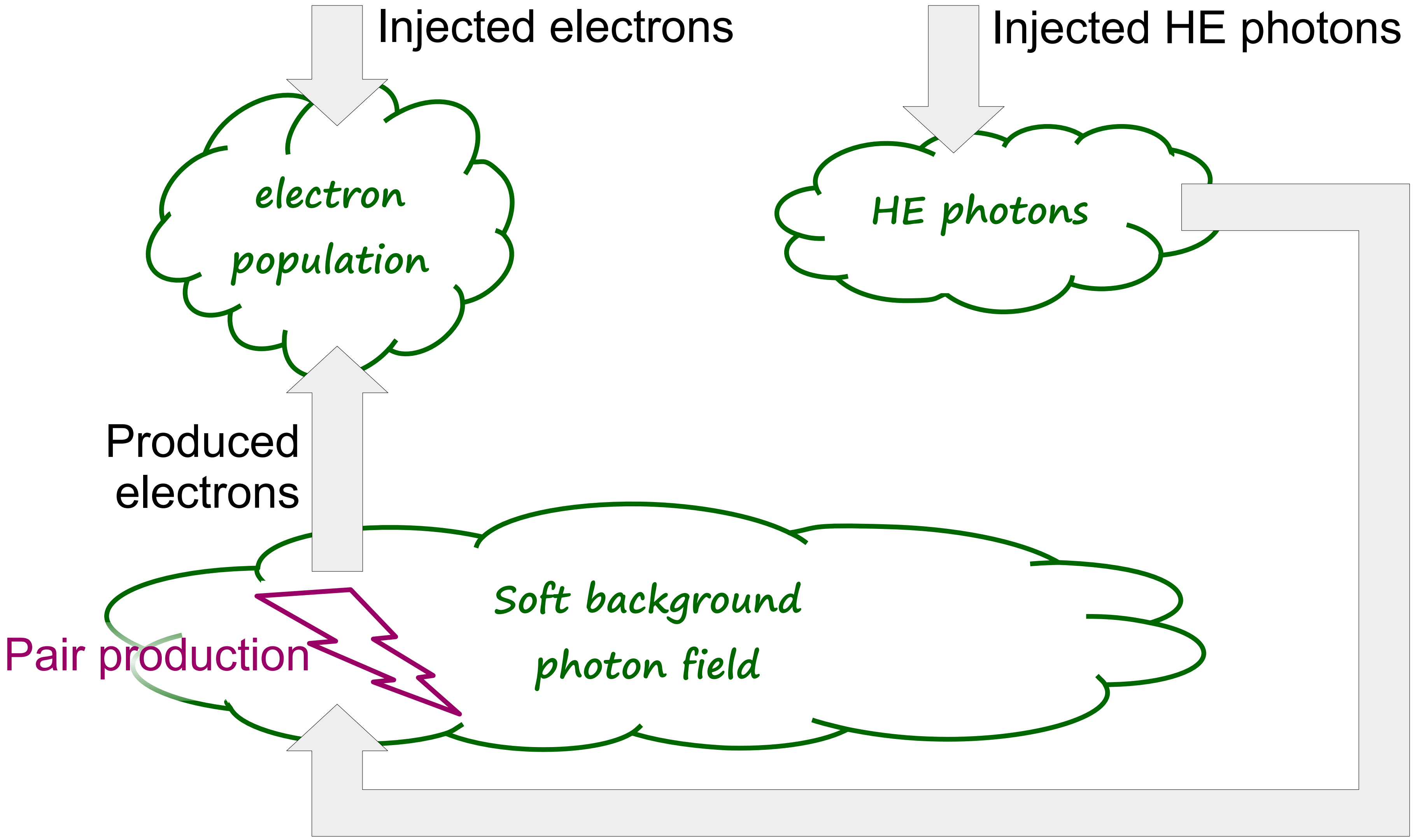
Cascades in AGN



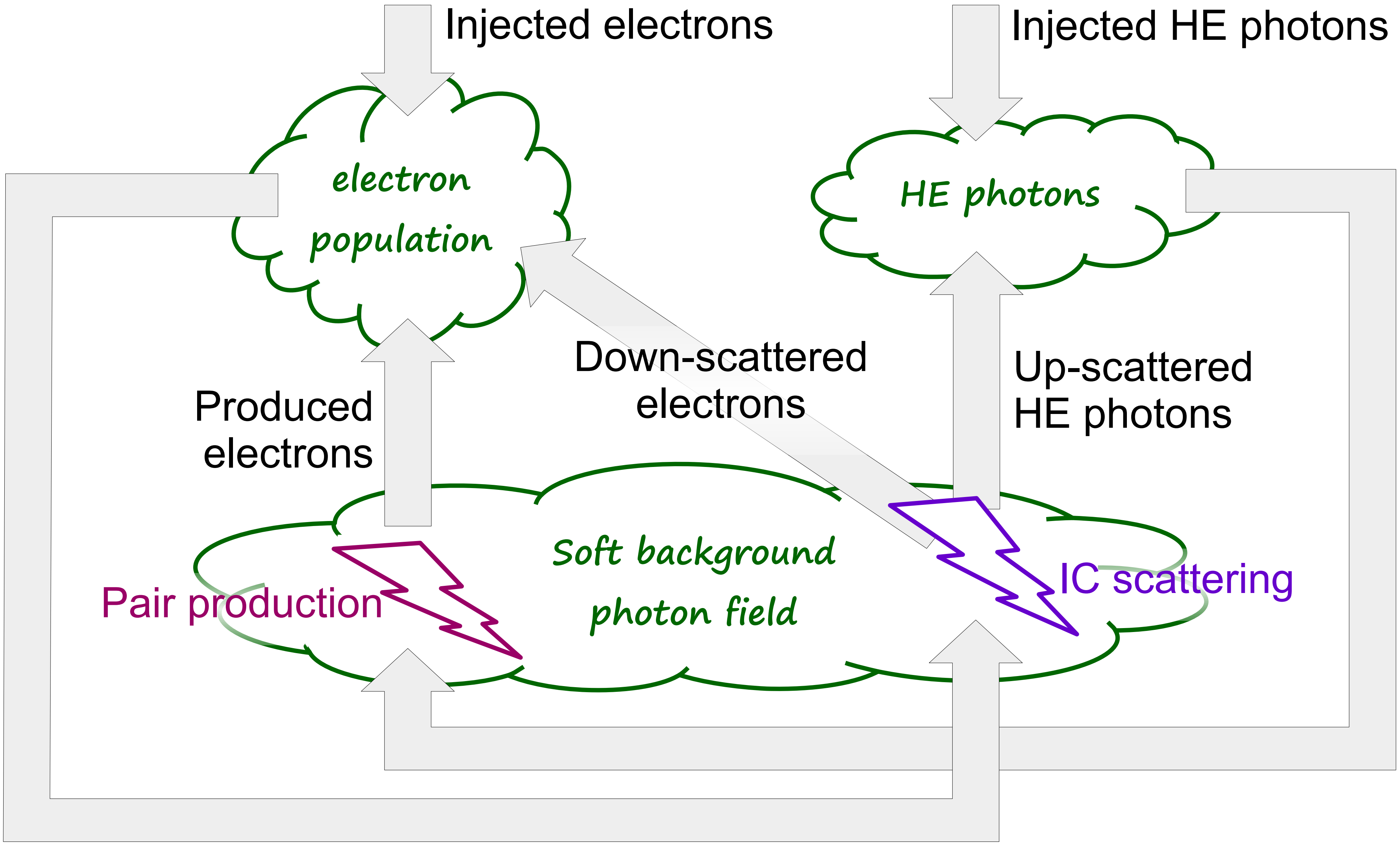
IC pair cascades



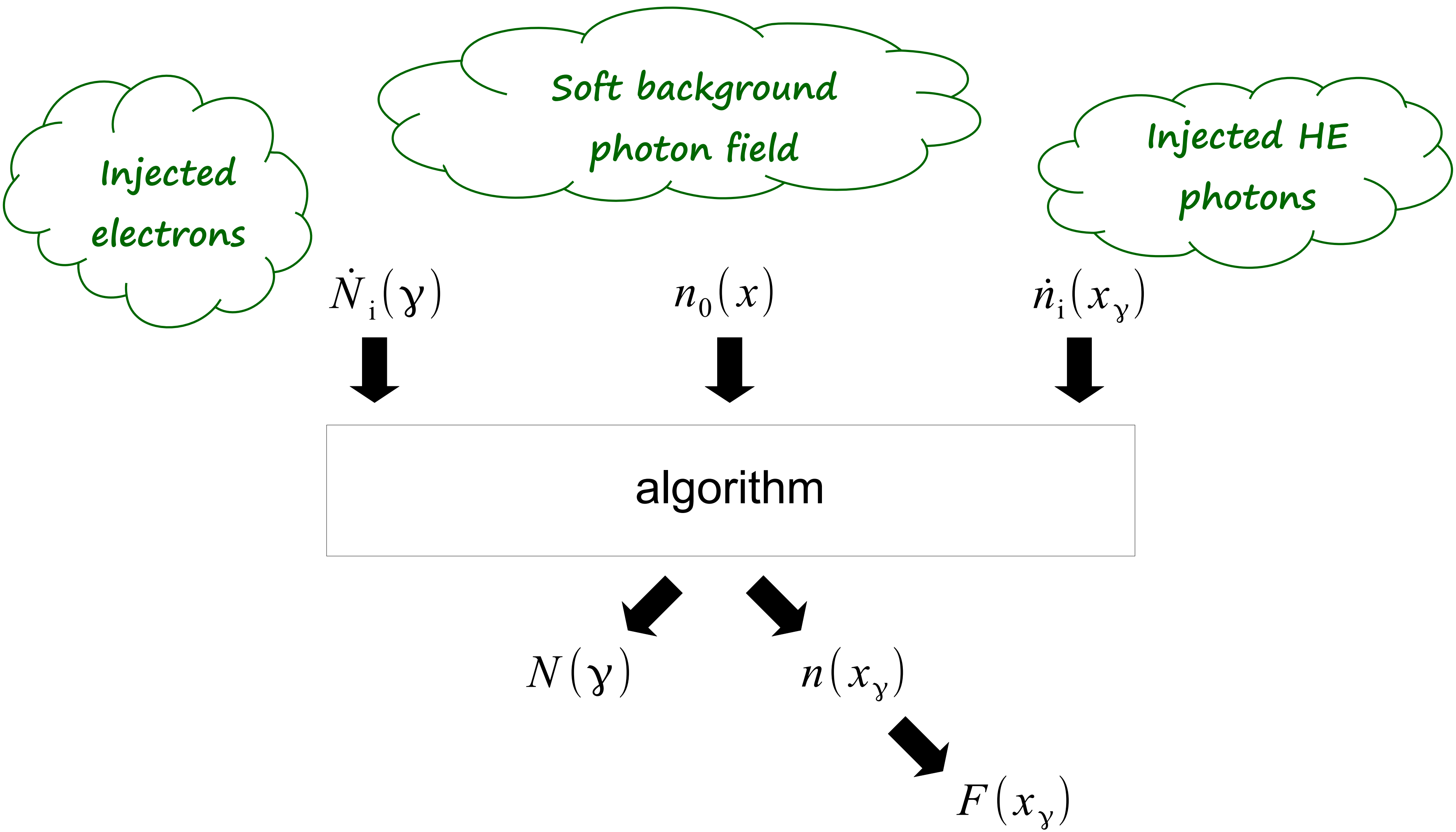
IC pair cascades



IC pair cascades



Python implementation



Python implementation

- Specify input quantities: $\dot{N}_{\mathrm{i}}(\gamma)$, $\dot{n}_{\mathrm{i}}(x_\gamma)$, $n_0(x)$
- Determine $C(\gamma,\gamma')$ and $p(x_\gamma,\gamma)$ via $n_0(x)$
- Determine electron distribution $N(\gamma)$
  - KN regime: Solve kinetic equation iteratively

$$N_j(\gamma)=\mathcal{F}\left(n_0,\dot{N}_{\mathrm{i}},\dot{n}_{\mathrm{i}},N_{j-1},\gamma\right)$$

  - Thomson regime: Integrate continuity equation

$$\frac{\mathrm{d}\left(\dot{\gamma}(\gamma)N(\gamma)\right)}{\mathrm{d}\gamma}=\dot{N}_{\mathrm{i}}(\gamma)+\dot{N}_{\mathrm{PP}}(\gamma)$$

- Determine HE photon spectrum $n(x_\gamma)$
- Convert to flux density $F(x_\gamma)$

Specify input quantities

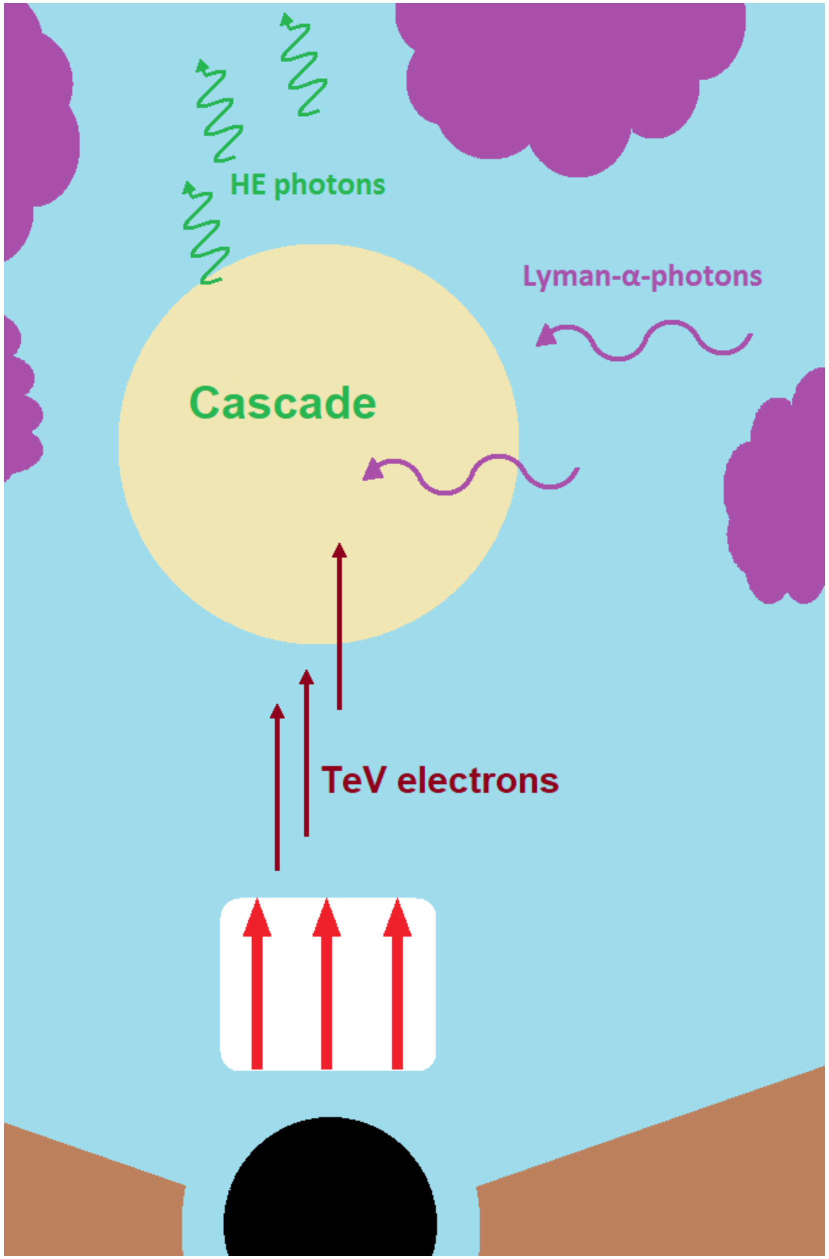
$$\dot{N}_{\mathrm{i}}(\gamma)=\begin{cases}\text{Gaussian around } \gamma_{\mathrm{mean}} & \text{if } \gamma_1\leq\gamma\leq\gamma_0\\ 0 & \text{otherwise}\end{cases}$$

$$\gamma_{\mathrm{mean}}\approx\mathrm{TeV}$$

$$n_0(x)=K_2\cdot\delta_{\mathrm{Dirac}}(x-x_0)$$

$$x_0=h/(121.5\,\mathrm{nm}\,m_{\mathrm{e}}\,c)$$

$$\dot{n}_{\mathrm{i}}(x_\gamma)=0$$



Python implementation

- Specify input quantities: $\dot{N}_\mathrm{i}(\gamma)$, $\dot{n}_\mathrm{i}(x_\gamma)$, $n_0(x)$
- Determine $C(\gamma,\gamma')$ and $p(x_\gamma,\gamma)$ via $n_0(x)$
- Determine electron distribution $N(\gamma)$
  - KN regime: Solve kinetic equation iteratively

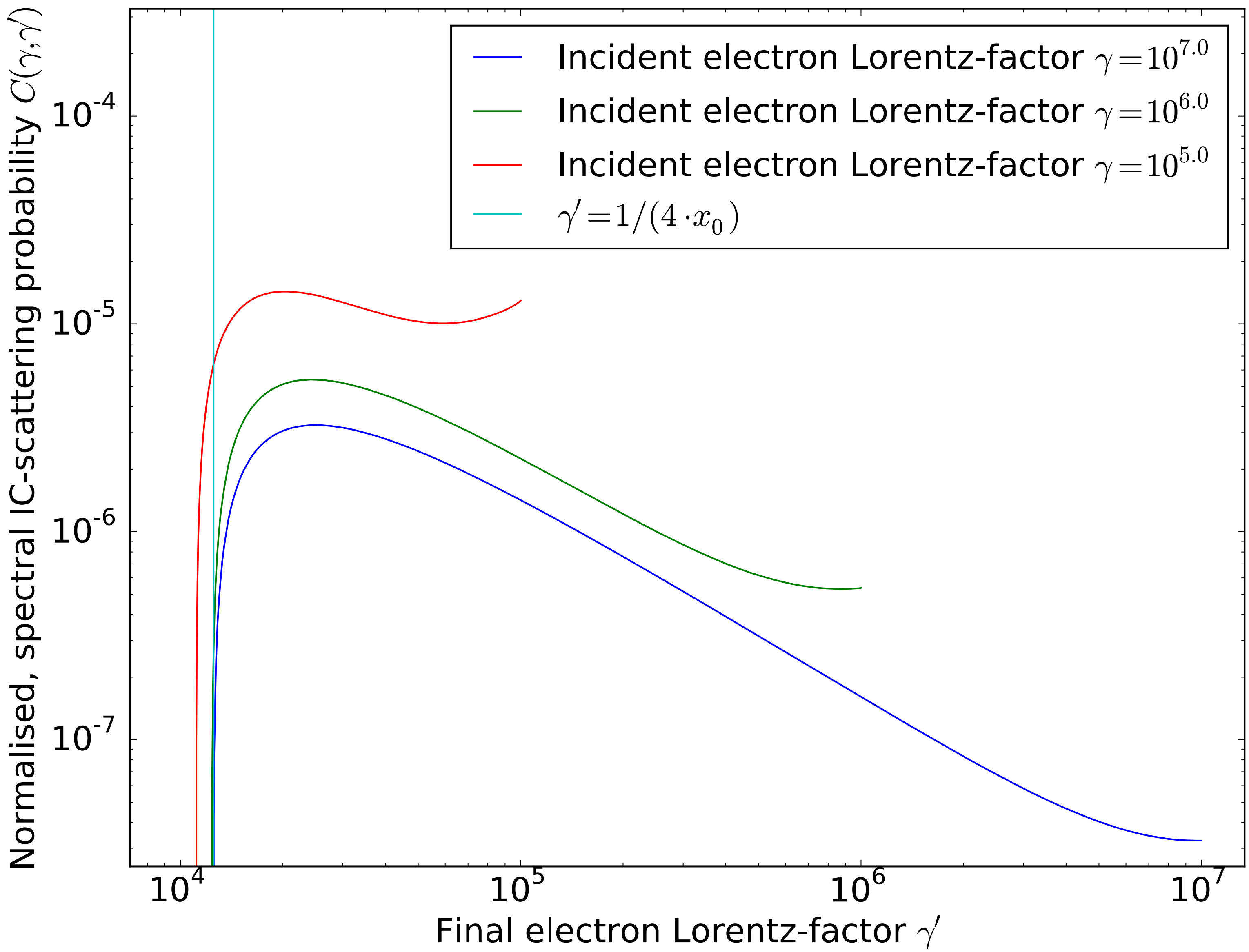
$$N_j(\gamma) = \mathcal{F}\left(n_0, \dot{N}_\mathrm{i}, \dot{n}_\mathrm{i}, N_{j-1}, \gamma\right)$$

  - Thomson regime: Integrate continuity equation

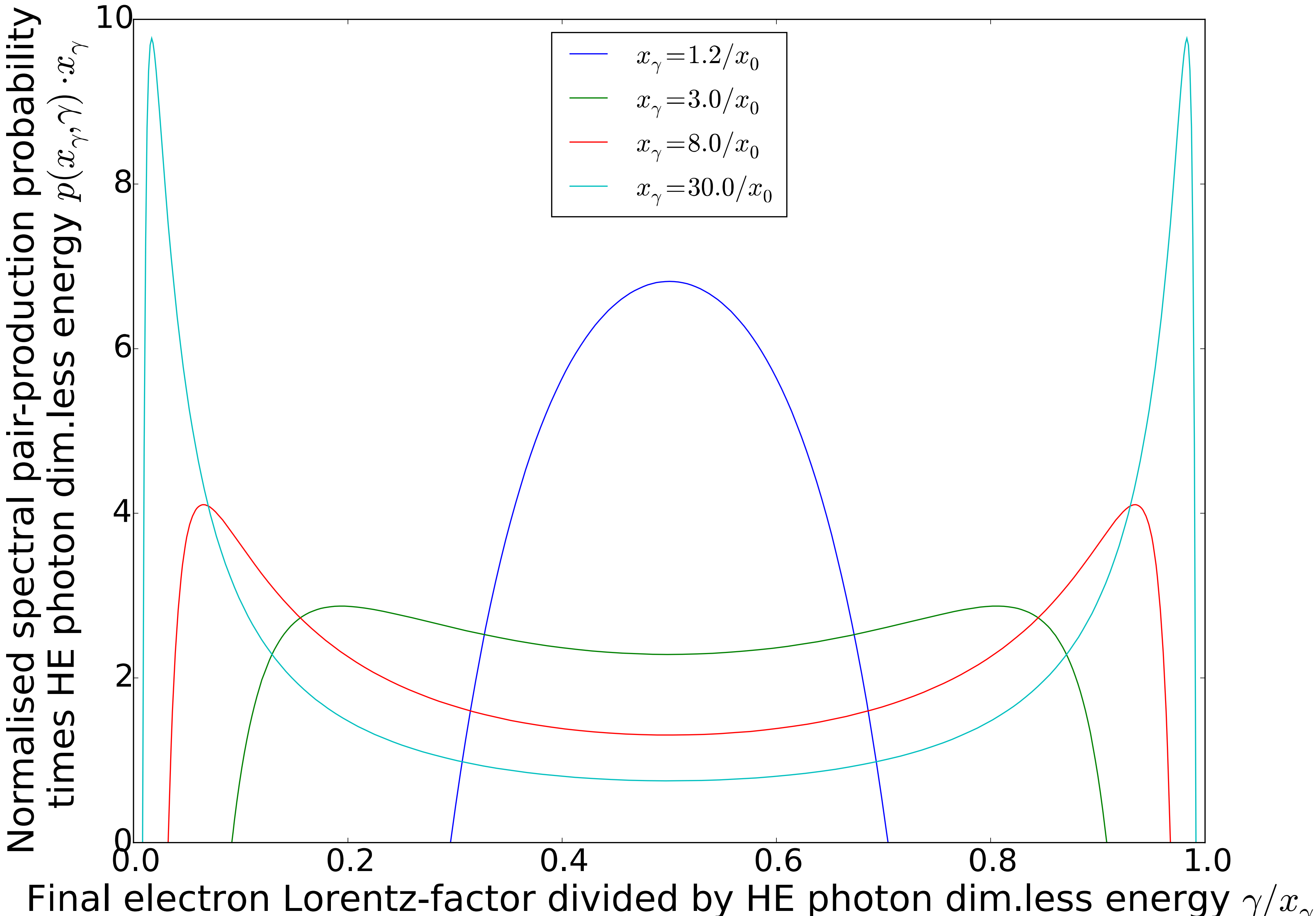
$$\frac{\mathrm{d}\left(\dot{\gamma}(\gamma)N(\gamma)\right)}{\mathrm{d}\gamma} = \dot{N}_\mathrm{i}(\gamma) + \dot{N}_\mathrm{PP}(\gamma)$$

- Determine HE photon spectrum $n(x_\gamma)$
- Convert to flux density $F(x_\gamma)$

Determine $C(\gamma, \gamma')$



Determine $p(x_\gamma, \gamma)$



Python implementation

- Specify input quantities: $\dot{N}_{\mathrm{i}}(\gamma)$, $\dot{n}_{\mathrm{i}}(x_\gamma)$, $n_0(x)$
- Determine $C(\gamma,\gamma')$ and $p(x_\gamma,\gamma)$ via $n_0(x)$
- Determine electron distribution $N(\gamma)$
  - KN regime: Solve kinetic equation iteratively

$$N_j(\gamma)=\mathscr{F}\left(n_0,\dot{N}_{\mathrm{i}},\dot{n}_{\mathrm{i}},N_{j-1},\gamma\right)$$

  - Thomson regime: Integrate continuity equation

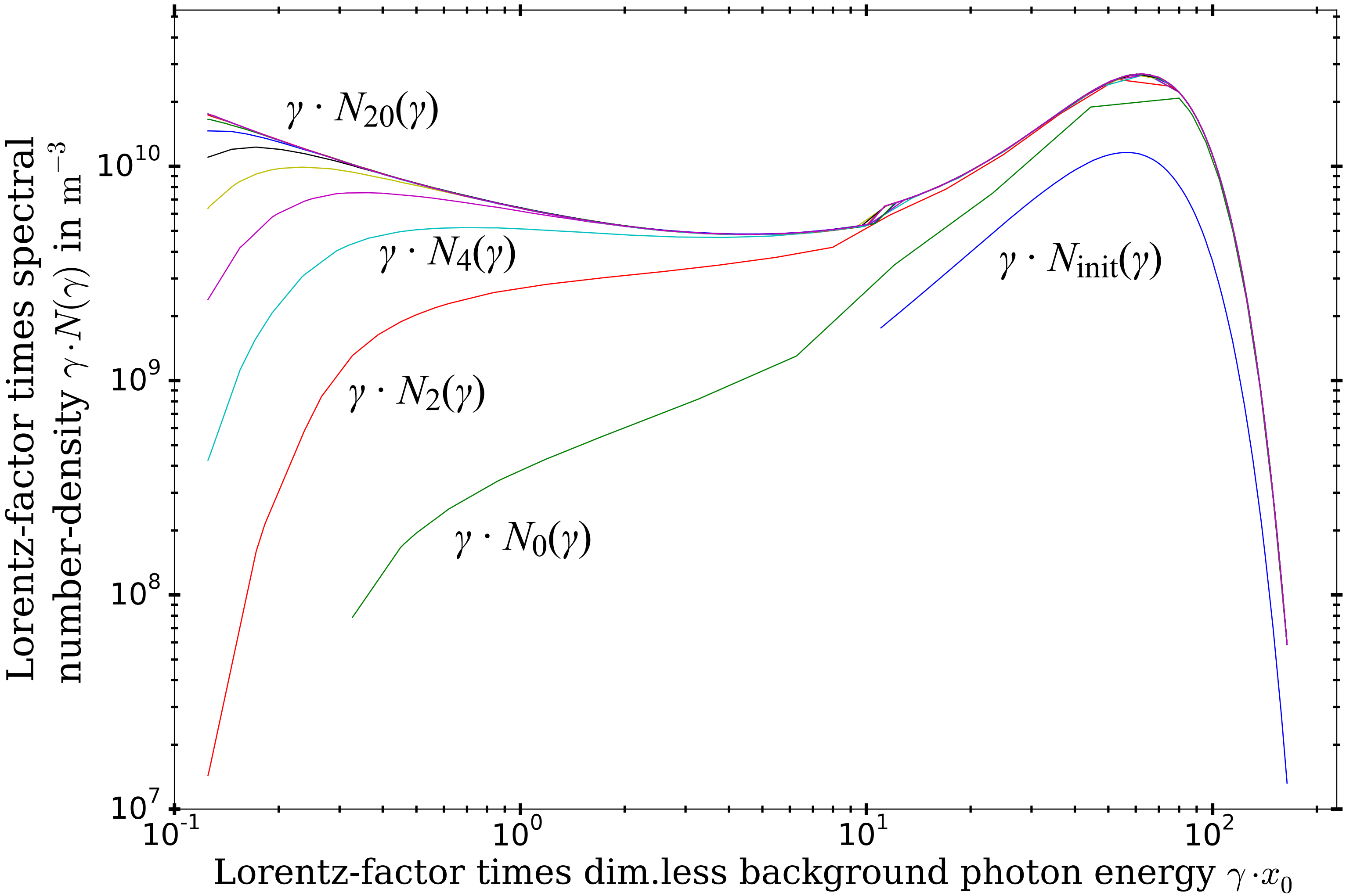
$$\frac{\mathrm{d}\left(\dot{\gamma}(\gamma)N(\gamma)\right)}{\mathrm{d}\gamma}=\dot{N}_{\mathrm{i}}(\gamma)+\dot{N}_{\mathrm{PP}}(\gamma)$$

- Determine HE photon spectrum $n(x_\gamma)$
- Convert to flux density $F(x_\gamma)$

Solve cascade equation

$$N_j(\gamma) = \mathscr{F}\left(n_0, \dot{N}_i, N_{j-1}, \gamma\right)$$

$\mathscr{F}$ from Zdziarski, 1988



Python implementation

- Specify input quantities: $\dot{N}_{\mathrm{i}}(\gamma)$, $\dot{n}_{\mathrm{i}}(x_\gamma)$, $n_0(x)$
- Determine $C(\gamma,\gamma')$ and $p(x_\gamma,\gamma)$ via $n_0(x)$
- Determine electron distribution $N(\gamma)$
  - KN regime: Solve kinetic equation iteratively

    $$N_j(\gamma)=\mathcal{F}\left(n_0,\dot{N}_{\mathrm{i}},\dot{n}_{\mathrm{i}},N_{j-1},\gamma\right)$$

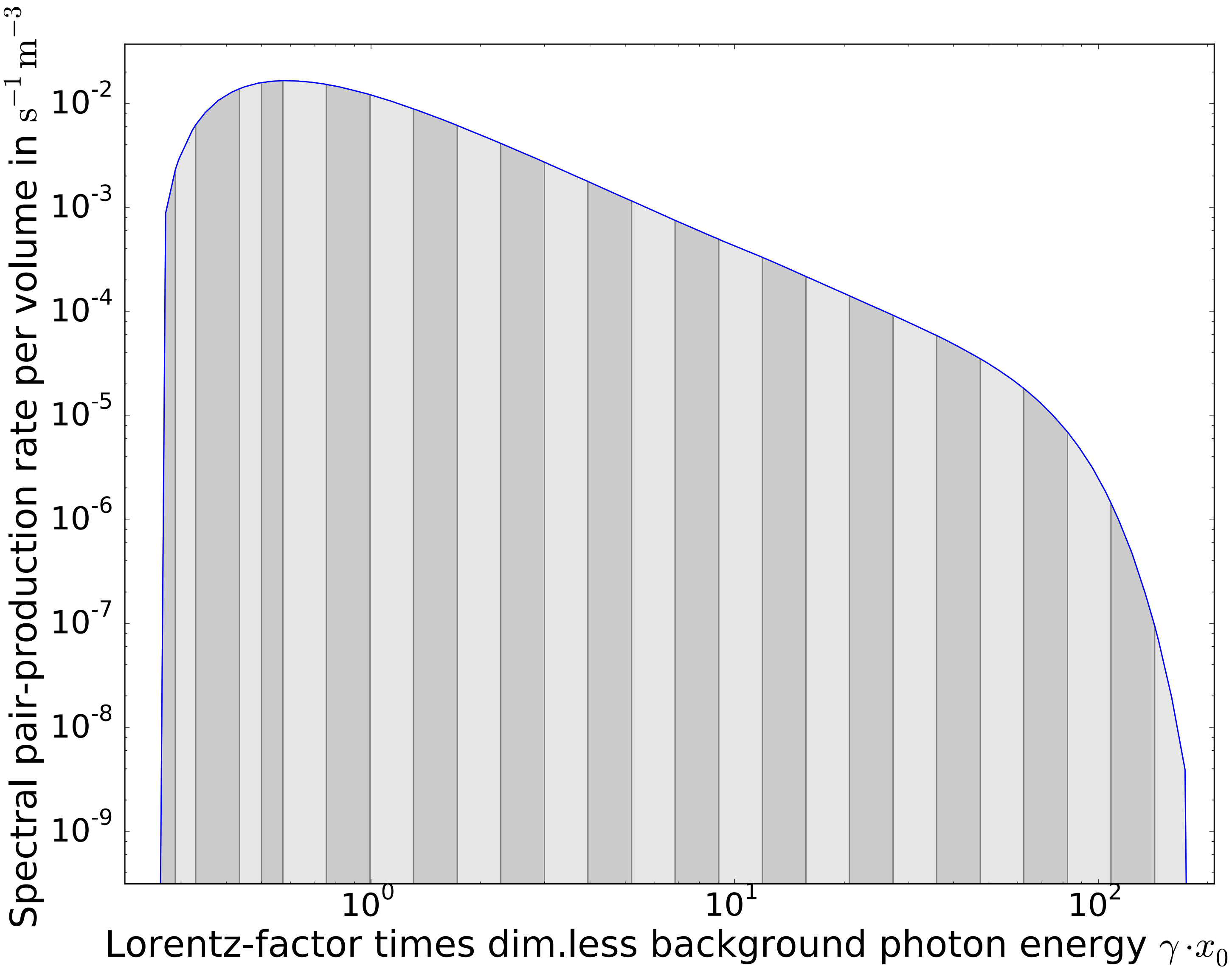
  - Thomson regime: Integrate continuity equation

    $$\frac{\mathrm{d}\left(\dot{\gamma}(\gamma)N(\gamma)\right)}{\mathrm{d}\gamma}=\dot{N}_{\mathrm{i}}(\gamma)+\dot{N}_{\mathrm{PP}}(\gamma)$$

- Determine HE photon spectrum $n(x_\gamma)$
- Convert to flux density $F(x_\gamma)$

Integrate continuity equation

$$\frac{\mathrm{d}\left(\dot{\gamma}(\gamma) N(\gamma)\right)}{\mathrm{d}\gamma} = \dot{N}_{\mathrm{i}}(\gamma) + \dot{N}_{\mathrm{PP}}(\gamma)$$



Python implementation

- Specify input quantities: $\dot{N}_{\mathrm{i}}(\gamma)$, $\dot{n}_{\mathrm{i}}(x_\gamma)$, $n_0(x)$
- Determine $C(\gamma,\gamma')$ and $p(x_\gamma,\gamma)$ via $n_0(x)$
- Determine electron distribution $N(\gamma)$
  - KN regime: Solve kinetic equation iteratively

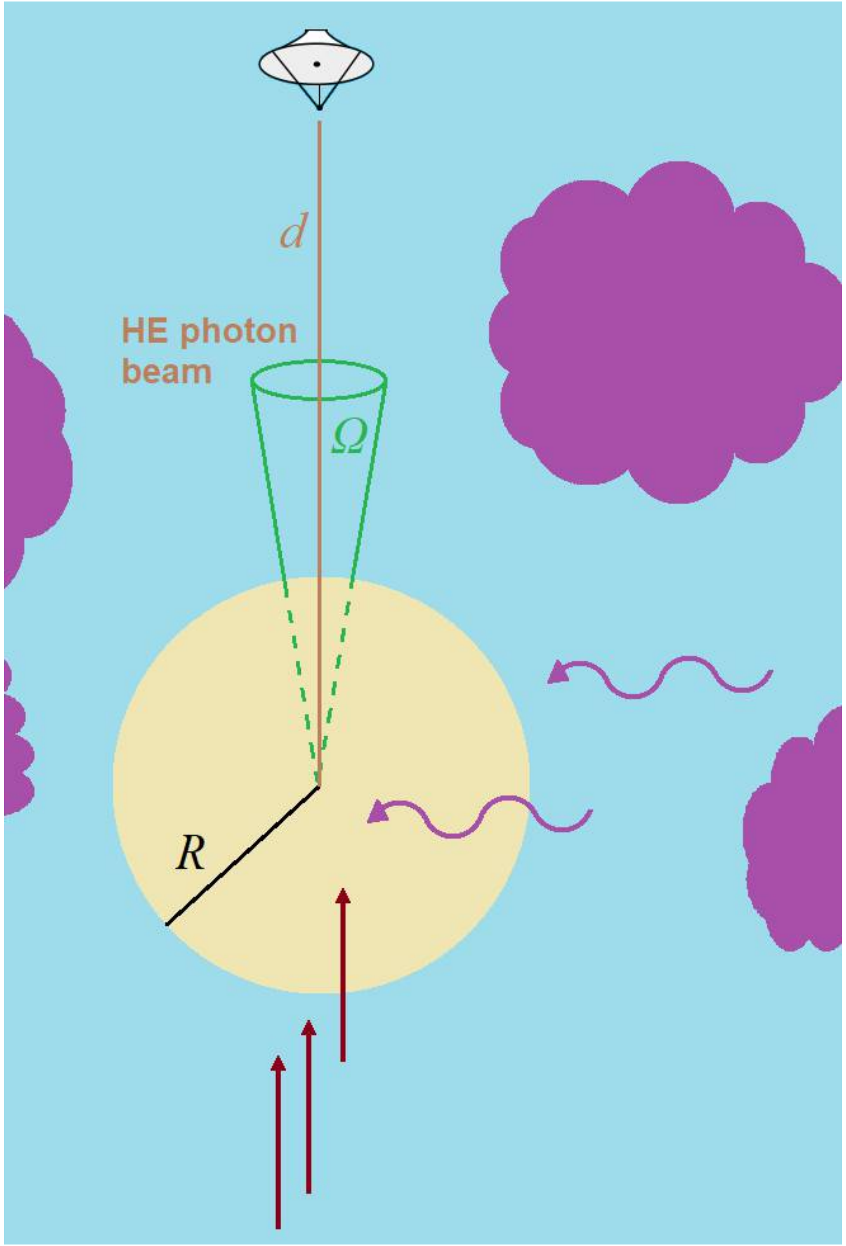
    $$N_j(\gamma)=\mathcal{F}\left(n_0,\dot{N}_{\mathrm{i}},\dot{n}_{\mathrm{i}},N_{j-1},\gamma\right)$$

  - Thomson regime: Integrate continuity equation

    $$\frac{\mathrm{d}\left(\dot{\gamma}(\gamma)N(\gamma)\right)}{\mathrm{d}\gamma}=\dot{N}_{\mathrm{i}}(\gamma)+\dot{N}_{\mathrm{PP}}(\gamma)$$

- Determine HE photon spectrum $n(x_\gamma)$
- Convert to flux density $F(x_\gamma)$

Determine HE photon spectrum



Spectral number density:

$$n(x_\gamma) = \mathscr{H}(n_0, \dot{n}_\mathrm{i}, N, x_\gamma)$$

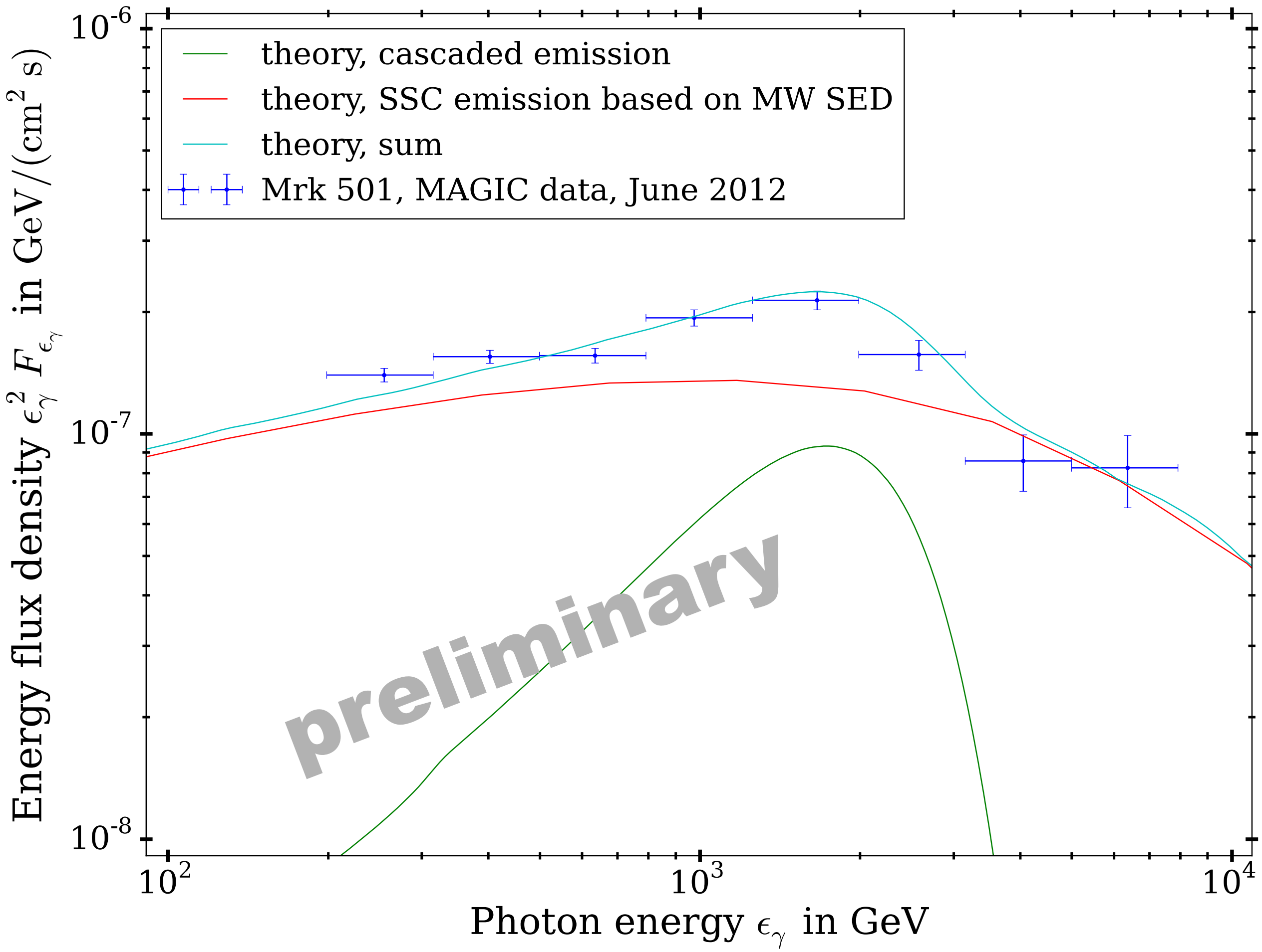
Convert to observed flux (νF_ν):

$$F(x_\gamma) = \frac{4\pi c R^2}{\Omega d^2} n(x_\gamma)$$

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

IC pair cascade in Mrk 501

ASTROPHYSIK



Quantity	Used value
K_1	$1.80 \cdot 10^3$ (s · m^3)$^{-1}$
K_2	$8 \cdot 10^{16}$ m^{-3}
γ_{mean}	$1.15 \cdot 10^{12}$ eV/ $(m_e c^2/e)$
σ	0.80 γ_{mean}
M	$5 \cdot 10^8$ $M_\odot$
R	1.0 r_s
Ω	0.001 sterad

- Spectral signature of gap activity
- VHE generation partly within inner portion of AGN

SSC modelling by Amit Shukla with code by H. Krawczynski et al. 2004.

Data processing by David Paneque et al. to be published

Similar work:

- Jones, 1968, PhRv 167, 1159
- Zdziarski, 1988, ApJ 335, 786
- Mannheim & Biermann, 1989, A&A 221, 221
- Levinson & Rieger, 2011, ApJ 730, 123
- Wendel et al., 2017, AIPC 1792, 26